MDPI

Article

Exploiting Generalized Cyclic Symmetry to Find Fast Rectangular Matrix Multiplication Algorithms Easier

Charlotte Vermeylen [1]*, Nico Vervliet [1,2], Lieven De Lathauwer [1,2], and Marc Van Barel [3]

[1] Department of Electrical Engineering (ESAT), KU Leuven, Kasteelpark Arenberg 10 bus 2246, Leuven B-3001, Belgium.
[2] Subfaculty Science and Technology, KULAK, E. Sabbelaan 53, Kortrijk B-8500, Belgium.
[3] Department of Computer Science, KU Leuven, Celestijnenlaan 200A, Leuven B-3001, Belgium.
* Correspondence: charlotte.vermeylen@kuleuven.be;

Abstract

The quest to multiply two large matrices as fast as possible is one that already intrigues researchers for several decades. However, the 'optimal' algorithm for a certain problem size is still not known. The fast matrix multiplication (FMM) problem can be formulated as a non-convex optimization problem—more specifically, as a challenging tensor decomposition problem. In this work, we build upon a state-of-the-art augmented Lagrangian algorithm, which formulates the FMM problem as a constrained least squares problem, by incorporating a new, generalized cyclic symmetric (CS) structure in the decomposition. This structure decreases the number of variables, thereby reducing the large search space and the computational cost per iteration. The constraints are used to find practical solutions, i.e., decompositions with simple coefficients, which yield fast algorithms when implemented in hardware. For the FMM problem, usually a very large number of starting points are necessary to converge to a solution. Extensive numerical experiments for different problem sizes demonstrate that including this structure yields more 'unique' practical decompositions for a fixed number of starting points. Uniqueness is defined relative to the known scale and trace invariance transformations that hold for all FMM decompositions. Making it easier to find practical decompositions may lead to the discovery of faster FMM algorithms when used in combination with sufficient computational power. Lastly, we show that the CS structure reduces the cost to multiply a matrix with itself.

Keywords: matrix multiplication; tensors; optimization; canonical polyadic decomposition; cyclic symmetry

Received:
Revised:
Accepted:
Published:

Citation: Vermeylen, C.; Vervliet, N.; De Lathauwer, L.; Van Barel, M. Exploiting generalized cyclic symmetry to find fast rectangular matrix multiplication algorithms easier. *Mathematics* **2025**, *1*, 0. https://doi.org/

Copyright: © 2025 by the authors. Submitted to *Mathematics* for possible open access publication under the terms and conditions of the Creative Commons Attri- bution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Matrix multiplication is a fundamental operation in numerical linear algebra, with applications ranging from machine learning to signal processing. Discovering faster algorithms for this task can therefore have wide-reaching impact. For very large systems, saving a few percent of computation time can be worth millions of dollars. The study of fast matrix multiplication (FMM) began with Strassen's breakthrough in 1969 [1], and despite decades of progress, many questions remain open. Recently, new algorithms have been discovered using deep reinforcement learning [2], while an augmented Lagrangian (AL) method has been proposed to obtain more stable and efficient decompositions [3]. Existing optimization-based approaches for discovering FMMs [2,4–8] often require significant computational resources and manual tuning, making them difficult to replicate. In this work, we propose a framework that does not require big computational resources and does

not require manual tinkering, enabling the discovery of FMM algorithms in a more efficient and reproducible manner.

The *fast matrix multiplication (FMM)* problem aims for faster ways to multiply large matrices by rewriting the bilinear equations of matrix multiplication as a tensor equation:

$$\mathbf{Z}(i,j) = \sum_{k=1}^{K} \mathbf{X}(i,k)\mathbf{Y}(k,j) \quad \rightarrow \quad \mathbf{Z}^\top = \sum_{i,j,k=1}^{I,J,K} \langle \mathbf{E}_{ik}^{I\times K}, \mathbf{X}\rangle_{\mathrm{F}} \langle \mathbf{E}_{kj}^{K\times J}, \mathbf{Y}\rangle_{\mathrm{F}} \mathbf{E}_{ji}^{J\times I} =: \left(\sum_{i,j,k=1}^{I,J,K} \mathbf{E}_{ik}^{I\times K} \otimes \mathbf{E}_{kj}^{K\times J} \otimes \mathbf{E}_{ji}^{J\times I} \right) \cdot_{\mathrm{F}} (\mathbf{X}, \mathbf{Y}), \tag{1}$$

where $\mathbf{Z} := \mathbf{X}\mathbf{Y}$ and $\mathbf{E}_{ij}^{I\times J}$, for $i = 1, \ldots, I$, and $j = 1, \ldots, J$, is a basis of matrices in $\mathbb{R}^{I\times J}$ such that $\mathbf{E}_{ij}^{I\times J}(i_1, j_2) = 1$, if $(i_1, j_2) = (i, j)$, and zero otherwise. '$\otimes$' denotes the tensor or outer product, and $\langle\cdot\rangle_{\mathrm{F}}$ and '$\cdot_{\mathrm{F}}$' indicate Frobenius inner products. A transpose is added in (1) such that tensor that is defined has additional interesting properties, such as *cyclic symmetry (CS)*, which will be discussed further in Section 2.3.2.

The rank of matrix multiplication is the minimal integer R such that

$$\mathbf{Z}^\top = \sum_{r=1}^{R} \langle \mathbf{U}_r, \mathbf{X}\rangle_{\mathrm{F}} \langle \mathbf{V}_r, \mathbf{Y}\rangle_{\mathrm{F}} \mathbf{W}_r, \tag{2}$$

for some matrices $\mathbf{U}_r \in \mathbb{R}^{I\times K}$, $\mathbf{V}_r \in \mathbb{R}^{K\times J}$, and $\mathbf{W}_r \in \mathbb{R}^{J\times I}$. It can be shown that minimizing R minimizes the computational complexity of matrix multiplication [9, Proposition 15.1], [10]. More specifically, the number of arithmetic operations needed to multiply two matrices in $\mathbb{R}^{I\times I}$ is $\mathcal{O}(I^\omega)$, where $\omega := \log_I R$. Since $R < I^3$, it holds that $\omega < 3$. Additionally, the exponent is bounded from below by two, since you need at least I^2 operations to compute as many elements.

Minimizing R in (2) corresponds to finding a *canonical polyadic decomposition (CPD)* of the *matrix multiplication tensor (MMT)* defined implicitly in (1) [1]:

$$\mathcal{T}_{IKJ} := \sum_{i,j,k=1}^{I,J,K} \mathbf{e}_{ik}^{IK} \otimes \mathbf{e}_{kj}^{KJ} \otimes \mathbf{e}_{ji}^{IJ}, \tag{3}$$

where $\mathcal{T}_{IKJ}$ has dimensions $IK \times KJ \times IJ$ and $\mathbf{e}_{ik}^{IK} := \mathrm{vec}\left(\mathbf{E}_{ik}^{I\times J}\right)$, where $\mathrm{vec}(\cdot)$ denotes vectorization along the columns. A MMT is a sparse tensor consisting of IJK ones and zeros elsewhere:

$$\mathcal{T}_{IKJ}(i_1 + (k_2-1)I, k_1 + (j_2-1)K, j_1 + (i_2-1)J) = \begin{cases} 1 & \text{if } i_1 = i_2, j_1 = j_2, k_1 = k_2, \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

for all $i_1, i_2 = 1, \ldots, I$, $j_1, j_2 = 1, \ldots, J$, and $k_1, k_2 = 1, \ldots, K$. A MMT thus only depends on the size of the matrices that are multiplied. A polyadic decomposition (PD) of $\mathcal{T}_{IKJ}$ of *length* R decomposes $\mathcal{T}_{IKJ}$ into R rank-1 tensors:

$$\mathcal{T}_{IKJ} = \sum_{r=1}^{R} \mathbf{u}_r \otimes \mathbf{v}_r \otimes \mathbf{w}_r, \tag{5}$$

where $\mathbf{u}_r, \mathbf{v}_r$, and $\mathbf{w}_r$ are vectors in $\mathbb{R}^{IK}$, $\mathbb{R}^{KJ}$, and $\mathbb{R}^{IJ}$ respectively. These vectors can be collected in three so-called *factor matrices*: $\mathbf{U} := [\mathbf{u}_1, \cdots, \mathbf{u}_R]$, $\mathbf{V} := [\mathbf{v}_1, \cdots, \mathbf{v}_R]$, and $\mathbf{W} := [\mathbf{w}_1, \cdots, \mathbf{w}_R]$. The rank R^* or $R^*(\mathcal{T})$ denotes the minimal R for which (5) holds. The decomposition is then called canonical (CPD). A PD with R terms is denoted by PD_R and

$\mathrm{PD}_R(\mathcal{T})$ denotes a PD_R of a certain tensor $\mathcal{T}$. The equations (4)-(5) are also known as the Brent equations [11].

Decompositions of MMTs yield *base algorithms* for FMM of the form

$$\mathrm{vec}\left(\mathbf{Z}^\top\right) = \sum_{r=1}^{R} \langle \mathbf{u}_r, \mathrm{vec}(\mathbf{X})\rangle \langle \mathbf{v}_r, \mathrm{vec}(\mathbf{Y})\rangle \mathbf{w}_r, \tag{6}$$

which is the same equation as (2) with $\mathbf{u}_r = \mathrm{vec}(\mathbf{U}_r)$, $\mathbf{v}_r = \mathrm{vec}(\mathbf{V}_r)$, and $\mathbf{w}_r = \mathrm{vec}(\mathbf{W}_r)$, for $r = 1, \ldots, R$. The total number of operations of a base algorithm is higher than for the standard algorithm. However, the *active* multiplications, i.e., the number of multiplications between (linear combinations of) elements of $\mathbf{X}$ and $\mathbf{Y}$, is reduced from IJK to R. Algorithm (6) becomes faster when applied sufficiently many times recursively to multiply large matrices. This is because the active multiplications determine the asymptotic complexity [12]. In this case, the large matrices, e.g., of size $I^l \times K^l$ and $K^l \times J^l$ for some $l \in \mathbb{N}$, are divided into $I \times K$ and $K \times J$ blocks of equal size, e.g., $I^{l-1} \times K^{l-1}$ and $K^{l-1} \times J^{l-1}$, respectively. All operations in (6) are then directly performed on the blocks. For example, the vectorization operator stacks the blocks in a block vector of length IK and JK, respectively, where each element is again a matrix. The inner products are calculated on these block vectors, where each scalar multiplication becomes a scalar element-wise multiplication on all elements of each block. Each multiplication between the inner products becomes another matrix multiplication of smaller dimension and can be further divided into smaller blocks if possible. A more detailed analysis of the recursive application of FMM algorithms is given in Section 2.2.

The rank of $\mathcal{T}_{IKJ}$, denoted by $R^*(\mathcal{T}_{IKJ})$, is not known for most problem parameters I, J, and K, which is one of the main difficulties of the FMM problem. One of the exceptions is $\mathcal{T}_{222}$, of which the rank is known to be seven [13] and the problem is fully understood. We call $\mathrm{PD}_7(\mathcal{T}_{222})$ discovered by Strassen $\mathrm{PD}_{\text{Strassen}}$ [1]. Only lower and upper bounds on the rank exist for most other I, J, and K. For example, it is known that $R^*(\mathcal{T}_{333})$ lies between 19 [14] and 23 [4,6,7,15]. The upper bound, i.e., the lowest rank of $\mathcal{T}_{IKJ}$ known, is denoted by $\tilde{R}(\mathcal{T}_{IKJ})$. The most recent overview table of $\tilde{R}(\mathcal{T}_{IKJ})$ is given in [16, Table 3]. However, since then, there has been some further improvement for $I = K = J = 4$ [17]. Note that permuting (I, J, K) does not change the rank. Recently, there has also been an interest in finding complex-valued decompositions of $\mathcal{T}_{IKJ}$ [16,18]. However, as the number of standard multiplications to multiply two complex numbers is at least three, the complexity is higher than for real decompositions of$\mathcal{T}_{IKJ}$. A recent overview of different techniques to obtain base algorithms for different dimensions is given in [19, Section 1.1].

To find a PD_R of $\mathcal{T}_{IKJ}$, the following *nonlinear least squares (NLS)* problem in $\mathbf{x}^\top := \begin{bmatrix}\mathrm{vec}(\mathbf{U})^\top & \mathrm{vec}(\mathbf{V})^\top & \mathrm{vec}(\mathbf{W})^\top\end{bmatrix}$ is formulated [4,5,7,11,20]:

$$\min_{\mathbf{x}} \underbrace{\frac{1}{2}\left\| \sum_{r=1}^{R} \mathbf{w}_r \otimes \mathbf{v}_r \otimes \mathbf{u}_r - \mathrm{vec}\left(\mathcal{T}_{IKJ}\right) \right\|^2}_{=:f(\mathbf{x})}, \tag{7}$$

where '$\otimes$' denotes the Kronecker product and $\|\cdot\|$ the ℓ_2-norm. Note that $\mathbf{x}^*$ is a PD_R of $\mathcal{T}_{IKJ}$ if and only if $f(\mathbf{x}^*) = 0$. However, state-of-the-art optimization algorithms are not guaranteed to converge to a global optimum of (7). Furthermore, the convergence can be slow, even for second-order methods. One of the reasons is that (7) is non-convex and PDs of MMT are not unique and have furthermore additional invariances compared to generic PDs [21]. This means that the minima of (7) are non-isolated.

In practice, the rank R^* is approximated experimentally by the lowest length $\tilde{R}$ for which a solution $\mathbf{x}^*$ with $f(\mathbf{x}^*) = 0$ can be obtained using numerical optimization. However, as no globally convergent algorithm for (7) exists, this does not mean that a solution with a lower rank doesn't exists as long as there is a gap with the theoretical lower bound.

Another difficulty is the existence of *border rank* PDs (BRPDs) [10,12]. BRPDs are ill-conditioned approximate decompositions of T_{IKJ} with elements that grow to infinity during convergence. The border rank of a particular tensor can be smaller than its canonical rank. This is closely related to the concept of degeneracy [22,23]. The BRPDs of MMTs are, e.g., investigated in [4,24–28]. To eliminate the convergence to BRPDs, constraints are usually added to (7) [3–5,8] or the elements are restricted to a discrete set of values [2,6]. The phenomenon of degeneracy also gives rise to regions of very slow convergence in the optimization landscape, informally known as *swamps* [29–31].

Additionally, the number of variables or unknowns grows rapidly with I, J, and K. Therefore, this paper proposes a new structure that can be enforced in PDs of MMTs to reduce the number of variables, which directly reduces the computational complexity of optimization algorithms to solve (7) and shrinks the large search space.

To solve (7), the alternating least squares (ALS) method is frequently used [4,7,20,32]. The local convergence of the ALS method for (7) is known under the assumption that the Hessian is positive definite at the solutions modulo the scaling indeterminacies [33]. This is however not satisfied for PDs of MMTs as they have additional invariances [21]. In [5] and [4], a constrained optimization problem and corresponding method are proposed to improve the convergence. More specifically, Lagrange multipliers and a quadratic penalty are used for the constraints, and a Levenberg–Marquardt (LM) and an ALS method, respectively, are used to solve the constrained problems. Still, these methods were unable to find PD_{49}s for 4×4 matrix multiplication, although they are known to exist. Another constrained optimization problem was proposed recently [3]:

$$\min_{\mathbf{x}} f(\mathbf{x}), \quad \text{s.t.} \quad h(\mathbf{x}) = 0, \quad l \leq \mathbf{x} \leq u, \tag{8}$$

together with an augmented Lagrangian (AL) method to solve it. The bound constraint ensures that the method does not converge to BRPDs. Different equality constraints $h(\mathbf{x})$ were proposed to obtain PDs with simple coefficient decompositions. The AL method has as advantage compared to adding quadratic penalty terms to (7) that even outside the neighborhood of an optimum, the constraints are satisfied accurately. New $\text{PD}_{49}(\mathcal{T}_{444})$s, and different new CS PDs are obtained using this method and the stability of existing algorithms is improved [3]. As inner optimization algorithm, the LM method is used, which is a (second-order) damped Gauss–Newton optimization algorithm that optimizes over all factor matrices simultaneously [34].

We use the constrained problem formulation (8) together with the AL method proposed in [3] to find new PDs of $\mathcal{T}_{IKJ}$ with the generalized CS structure proposed in this paper. The structure is imposed to find new decompositions more easily by reducing the search space and computational complexity per iteration, which can then, in turn, be used to reduce the complexity of matrix multiplication. Although it is not known whether for all I, J, and K, a CPD of $\mathcal{T}_{IKJ}$ with the proposed structure exists, the structure is very flexible, as it is determined by two parameters that can be varied and, for example, can be made smaller to be less restrictive. So far, we have not encountered any CPDs of $\mathcal{T}_{IKJ}$ of small dimensions that do not admit this structure for any combination of these two parameters.

1.1. Contribution

A generalization of the CS structure investigated in [7,20,35] to non-square matrix multiplication is proposed for decompositions of MMTs. The advantage of investigating this structure is threefold.

Firstly, this structure reduces the number of variables, thereby shrinking the large search space and speeding up the computation time per iteration when used in an optimization algorithm. We have implemented the structure efficiently in a state-of-the-art all-at-once optimization algorithm for the FMM problem and use a constrained problem formulation to find new practical solutions automatically without manual tinkering. A large number of numerical experiments illustrate that including this structure helps find more practical solutions for a fixed number of starting points.

Secondly, we prove that by exploiting this structure, the cost to multiply a matrix with itself can be reduced. We give an example with Strassen's decomposition, which shows that the CS structure enables to speed-up standard matrix multiplication already from matrices of size $I = J = K = 2^8 = 256$ instead of $2^{10} = 1024$.

Lastly, 'unique' decompositions of MMTs, based on the well-known invariance transformations [21], are of theoretical interest. Different classes of decompositions have already been investigated before [2,3,7,20,36], but not for rectangular fast matrix multiplication. To determine if two decompositions are equivalent, a non-trivial optimization problem arises. Using the result from [3], where it was shown that the rank of the Jacobian matrix is also invariant under these transformations, we use the different ranks to distinguish 'unique' practical decompositions.

1.2. Organization

The paper is organized as follows. Section 2 provides the preliminaries related to the FMM problem. In Section 3, the new CS structure is proposed and its advantages are discussed. Section 4 presents a large number of numerical experiments for different MMTs, illustrating that incorporating this structure helps to find more (unique) practical decompositions per fixed number of starting points.

2. Preliminaries

This section first discusses the notation and afterwards what is meant with practical decompositions and FMM algorithms. Section 2.3 gives a summary of the most important properties of decompositions of MMTs: invariances, cyclic symmetry, and recursive decompositions.

2.1. Notation

Scalars are denoted by lowercase letters, vectors by bold letters, matrices by bold uppercase letters, and tensors by calligraphic letters, e.g., a, $\mathbf{a}$, $\mathbf{A}$, and $\mathcal{A}$. The ith column of $\mathbf{A}$ is denoted by $\mathbf{a}_i$ and the jth element of $\mathbf{a}$ by a_j. '$\otimes$' denotes the tensor or outer product, '$\otimes$' the Kronecker product, $*$ the Hadamard or element-wise product, and $\text{vec}(\cdot)$ the vectorization operator. Vectorization of a tensor is performed such that $\text{vec}(\mathcal{T})(i + (j-1)J + (k-1)JK) = \mathcal{T}(i,j,k)$. $\mathbf{I}$ denotes the identity matrix of the appropriate size.

2.2. Practical algorithms

To multiply larger matrices, e.g., of size $I^l \times K^l$ and $K^l \times J^l$, a base algorithm for (I, J, K) matrix multiplication can be applied recursively, which requires $\mathcal{O}\left(cR^l\right)$ floating-point operations. The value of c depends on how many nonzero elements appear in the decomposition. When c and l are sufficiently small and large, respectively, the computational complexity of standard matrix multiplication, i.e., $\mathcal{O}\left((IKJ)^l\right)$ operations, can be lowered.

The following result is a generalization of the operation count of Strassen's decomposition in [1] to general FMM decompositions.

Proposition 1 (cost fast matrix multiplication)**.** *Let* $\mathbf{U}$, $\mathbf{V}$, *and* $\mathbf{W}$ *be the factor matrices of a base algorithm for* (I, K, J) *matrix multiplication. The complexity to use this base algorithm* l *times recursively to multiply two matrices* $X \in \mathbb{R}^{I^l \times K^l}$ *and* $Y \in \mathbb{R}^{K^l \times J^l}$ *is*

$$\left(1 + c_{\mathbf{U}} \frac{1}{R - IK} + c_{\mathbf{V}} \frac{1}{R - KJ} + c_{\mathbf{W}} \frac{1}{R - IJ}\right) R^l - c_{\mathbf{U}} \frac{(IK)^l}{R - IK} - c_{\mathbf{V}} \frac{(KJ)^l}{R - KJ} - c_{\mathbf{W}} \frac{(IJ)^l}{R - IJ}, \quad (9)$$

where $c_{\mathbf{U}}$ *and* $c_{\mathbf{V}}$ *are the number of additions and scalar multiplications needed to compute the inner products with* $\mathbf{u}_r$ *and* $\mathbf{v}_r$ *in* (6)*, respectively, and* $c_{\mathbf{W}}$ *is the number of additions and scalar multiplications with the* $\mathbf{w}_r$*.*

Proof. A base algorithm (6) of rank R of $\mathcal{T}_{IKJ}$ can be applied on the $I^{l-1} \times I^{l-1}$ submatrices of $\mathbf{X}$ and the $K^{l-1} \times J^{l-1}$ submatrices of $\mathbf{Y}$, where each active multiplication is then again a multiplication of $\left(I^{l-1}, K^{l-1}, J^{l-1}\right)$. The cost to multiply $\mathbf{X}$ and $\mathbf{Y}$ can thus be written as

$$\text{cost}\left(I^l, K^l, J^l\right) = R \cdot \text{cost}\left(I^{l-1}, K^{l-1}, J^{l-1}\right) + c_{\mathbf{U}} \cdot (IK)^{l-1} + c_{\mathbf{V}} \cdot (KJ)^{l-1} + c_{\mathbf{W}} \cdot (IJ)^{l-1}.$$

Applying this formula recursively until $\text{cost}(1, 1, 1) = 1$, leads to

$$\begin{aligned}
\text{cost}\left(I^l, K^l, J^l\right) &= R^l + \sum_{i=1}^{l} R^{i-1} \left(c_{\mathbf{U}} \cdot (IK)^{l-i} + c_{\mathbf{V}} \cdot (KJ)^{l-i} + c_{\mathbf{W}} \cdot (IJ)^{l-i}\right) \\
&= R^l + c_{\mathbf{U}} (IK)^{l-1} \sum_{i=0}^{l-1} \left(\frac{R}{IK}\right)^i + c_{\mathbf{V}} (KJ)^{l-1} \sum_{i=0}^{l-1} \left(\frac{R}{KJ}\right)^i + c_{\mathbf{W}} (IJ)^{l-1} \sum_{i=0}^{l-1} \left(\frac{R}{IJ}\right)^i \\
&= R^l + c_{\mathbf{U}} \frac{R^l - (IK)^l}{R - IK} + c_{\mathbf{V}} \frac{R^l - (KJ)^l}{R - KJ} + c_{\mathbf{W}} \frac{R^l - (IJ)^l}{R - IJ}
\end{aligned}$$

□

Remark: The constants $c_{\mathbf{U}}$, $c_{\mathbf{V}}$, and $c_{\mathbf{W}}$ depend implicitly on the parameters I, J, and K in the sense that each factor matrix contains at least IJK and at most IKR, JKR, and IJR nonzero elements, respectively. Furthermore, we can assume that each factor vector contains at least one nonzero, and the rows of $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{W}$ contain at least I, J, and K nonzeros each, respectively. This can be derived from the specific locations of the ones in the tensor as given in (4). The constants $c_{\mathbf{U}}$, $c_{\mathbf{V}}$, and $c_{\mathbf{W}}$ are directly related to the number of nonzeros, but the exact relation depends on the specific distribution of the nonzeros and their values.

The constants $c_{\mathbf{U}}$, $c_{\mathbf{V}}$, and $c_{\mathbf{W}}$ can be decreased by using sparse PDs with elements that are powers of 2 because multiplication with a power of 2 is not a costly operation when implemented in hardware. Such PDs are called *practical* because without small constants, standard matrix multiplication is only improved asymptotically.

A disadvantage of formulation (7) is that the solutions have floating-point elements. Consequently, if a numerical PD of $\mathcal{T}_{IKJ}$ is obtained, it has to be transformed to a practical one. The well-known *inv-transformations* [21] can be used for this purpose [5,8]. However, not all numerical PDs of $\mathcal{T}_{IKJ}$ can be transformed to a practical one [36]. Furthermore, this is not an easy process and involves a lot of manual tinkering [5,8]. In [36], this process is called *discretization*. Another possibility is to add constraints to the optimization problem, such as the following simple equality constraint proposed in [3]:

$$h_{\text{discr}}(\mathbf{x}) := \mathbf{x} * (\mathbf{x} + \mathbf{1}) * (\mathbf{x} - \mathbf{1}), \quad (10)$$

where '$*$' denotes element-wise multiplication. This constraint forces the elements to the discrete set $\{0, \pm 1\}$.

2.3. Properties of decompositions of matrix multiplication tensors

This section first briefly discusses the invariances for decompositions of $\mathcal{T}_{IKJ}$. Afterwards, other properties such as cyclic symmetry and recursive PDs are discussed in more detail.

2.3.1. Invariance transformations

Factor matrices are a non-unique representation of a PD. For example, the same tensor is obtained under permutation of the rank-1 tensors. Additionally, the factor vectors can be scaled as: $\alpha_r \mathbf{u}_r, \beta_r \mathbf{v}_r, \frac{1}{\alpha_r \beta_r} \mathbf{w}_r$, for all $r = 1, \ldots, R$, and α_r, β_r in $\mathbb{R}_0$.

Decompositions of $\mathcal{T}_{IKJ}$ also have other invariances [21]. See, e.g., [3] for an overview. The combination of the trivial transformations and the specific transformations that hold for decompositions of $\mathcal{T}_{IKJ}$ are called *inv-transformations*. They have dimension $I^2 + K^2 + J^2 + 2R - 3$. Two PDs of $\mathcal{T}_{IKJ}$ that can be transformed into one another in this way are called *inv-equivalent*. All $\text{PD}_7(\mathcal{T}_{222})$ are known to be inv-equivalent [13]. Some decompositions of $\mathcal{T}_{IKJ}$ also have additional invariances, see, e.g., [3].

The Jacobian matrix can be used to investigate the inv-equivalence of decompositions of MMTs and to discover additional invariances [3]. That is why in Section 4, the size and ranks of the Jacobian matrix at solutions with the proposed structure, are given.

2.3.2. Cyclic symmetry

Because of (3), $\mathcal{T}_{IKJ}$ is a structured tensor. More specifically, when $I = J = K$, $\mathcal{T}_{IKJ}$ is so-called cyclic symmetric (CS), which means that $\mathcal{T}_{III}(k,i,j) = \mathcal{T}_{III}(j,k,i) = \mathcal{T}_{III}(i,j,k)$, for $i, j, k = 1, \ldots, I^2$. This structure can be exploited in the factor matrices. More specifically, the rank-1 tensors can be forced to either occur in CS pairs or to be symmetric [35]:

$$\mathcal{T}_{III} = \sum_{s=1}^{S} \mathbf{a}_s \otimes \mathbf{a}_s \otimes \mathbf{a}_s + \sum_{t=1}^{T} (\mathbf{b}_t \otimes \mathbf{c}_t \otimes \mathbf{d}_t + \mathbf{c}_t \otimes \mathbf{d}_t \otimes \mathbf{b}_t + \mathbf{d}_t \otimes \mathbf{b}_t \otimes \mathbf{c}_t),$$

where $\mathbf{a}_s, \mathbf{b}_t, \mathbf{c}_t$, and $\mathbf{d}_t$ are in $\mathbb{R}^{I^2}$, for $s = 1, \ldots, S$ and $t = 1, \ldots, T$. Here, S and T count the symmetric rank-1 tensors and the CS rank-1 tensors, respectively. The length of a CS PD equals $R = 3T + S$. The matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$, can be defined containing the vectors $\mathbf{a}_s, \mathbf{b}_t, \mathbf{c}_t$, and $\mathbf{d}_t$ as columns and the factor matrices can be parameterized as: $\mathbf{U} = \begin{bmatrix} \mathbf{A} & \mathbf{B} & \mathbf{C} & \mathbf{D} \end{bmatrix}$, $\mathbf{V} = \begin{bmatrix} \mathbf{A} & \mathbf{D} & \mathbf{B} & \mathbf{C} \end{bmatrix}$, and $\mathbf{W} = \begin{bmatrix} \mathbf{A} & \mathbf{C} & \mathbf{D} & \mathbf{B} \end{bmatrix}$. The number of variables is only a third of the original value, making this parameterization highly useful for the FMM problem. Furthermore, the cost function can be changed to

$$\begin{aligned} \min_{\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D}} \frac{1}{2} \Bigg(& \sum_{i=1}^{I^2} \sum_{j=i}^{I^2} \sum_{k=i+1}^{I^2} \Bigg(\mathcal{T}_{III}(i,j,k) - \sum_{s=1}^{S} a_{is} a_{js} a_{ks} \\ & - \sum_{t=1}^{T} \left(b_{it} d_{jt} c_{kt} + c_{it} b_{jt} d_{kt} + d_{it} c_{jt} b_{kt} \right) \Bigg)^2 \\ & + \sum_{i=1}^{I^2} \left(\mathcal{T}_{III}(i,i,i) - \sum_{s=1}^{S} a_{is}^3 - 3 \sum_{t=1}^{T} \left(b_{it} d_{it} c_{it} \right) \right)^2 \Bigg), \end{aligned} \tag{11}$$

where $a_{is} := A(i,s)$, $b_{it} := B(i,t)$ and similarly for c_{it} and d_{it}, which reduces the number of rows in the Jacobian matrix used in NLS optimization methods to solve (11) from I^6 to $\frac{1}{3}(I^6 - I^2) + I^2$. Note that this cost function only includes one element for each CS pair of points because the CS structure already ensures that these elements are equal.

Different practical CS decompositions of rank 7 and 23 of $\mathcal{T}_{222}$ and $\mathcal{T}_{333}$, respectively, are known [7,20]. They were discovered through an ALS method and analyzed using group theory and algebraic geometry. Concerning Strassen's decomposition, $S = 1$ and $T = 2$. Although all $\mathrm{PD}_7(\mathcal{T}_{222})$s are known to be inv-equivalent, Strassen's decomposition can be transformed into another practical PD with parameters $(S, T) = (4, 1)$. However, this increases the number of nonzeros and is thus not used in practice.

2.4. Decompositions obtained by recursion

As mentioned in Section 2.2, an FMM algorithm applies a base algorithm for small I, J, and K, to large matrices, e.g., of size $I^l \times K^l$ and $K^l \times J^l$. In the same way, we can also obtain PDs of rank 49 of $\mathcal{T}_{444}$ using a $\mathrm{PD}_7(\mathcal{T}_{222})$ one time recursively. The $\mathrm{PD}_{49}(\mathcal{T}_{444})$ that is obtained using $\mathrm{PD}_{\text{Strassen}}$ is called $\mathrm{PD}^{\text{rec}}_{\text{Strassen}}$ in the rest of the text. In the following proposition, the formulas to obtain the factor matrices of a recursive PD from the factor matrices of the original PD are given. This result is well known, e.g., from the supplementary material of [2], and in [37], and a proof can be found, e.g., in [38, Section 2.2.3].

Proposition 2 (Recursive PD). *If* $\mathbf{U}$, $\mathbf{V}$, *and* $\mathbf{W}$ *are the factor matrices of a* $\mathrm{PD}_R(\mathcal{T}_{IKJ})$, *then the factor matrices* $\mathbf{U}'$, $\mathbf{V}'$, *and* $\mathbf{W}'$, *where*

$$\mathbf{u}'_r := \mathrm{vec}(\mathbf{U}_{r_1} \otimes \mathbf{U}_{r_2}), \qquad \mathbf{v}'_r := \mathrm{vec}(\mathbf{V}_{r_1} \otimes \mathbf{V}_{r_2}), \tag{12}$$

$$\mathbf{w}'_r := \mathrm{vec}(\mathbf{W}_{r_1} \otimes \mathbf{W}_{r_2}), \qquad r := r_2 + (r_1 - 1)R, \tag{13}$$

where '$\otimes$*' denotes the Kronecker product, for* $r_1, r_2 = 1, \ldots, R$, *are the factor matrices of a* $\mathrm{PD}_{R^2}\left(\mathcal{T}_{I^2K^2J^2}\right)$.

The following corollary gives the CS parameters and factor matrices of a recursive PD as a function of the original PD.

Corollary 3. *If* $\mathbf{U}$, $\mathbf{V}$, *and* $\mathbf{W}$ *are CS factor matrices of a* $\mathrm{PD}_R(\mathcal{T}_{III})$ *with CS parameters* S *and* T *such that* $\mathbf{U} = [\mathbf{A}\ \mathbf{B}\ \mathbf{C}\ \mathbf{D}]$, $\mathbf{V} = [\mathbf{A}\ \mathbf{D}\ \mathbf{B}\ \mathbf{C}]$, *and* $\mathbf{W} = [\mathbf{A}\ \mathbf{C}\ \mathbf{D}\ \mathbf{B}]$, *where* $\mathbf{A} \in \mathbb{R}^{I^2 \times S}$ *and* $\mathbf{B}, \mathbf{C}, \mathbf{D} \in \mathbb{R}^{I^2 \times T}$, *then the factor matrices* $\mathbf{U}'$, $\mathbf{V}'$, *and* $\mathbf{W}'$ *of the recursive decomposition* $\mathrm{PD}^{\text{rec}}_{R^2}(\mathcal{T}_{I^2I^2I^2})$ *is also CS with parameters* $S' = S^2$ *and* $T' = T(S + R)$:

$$\mathbf{U}' := \begin{bmatrix}\mathbf{A}' & \mathbf{B}' & \mathbf{C}' & \mathbf{D}'\end{bmatrix}, \qquad \mathbf{V}' := \begin{bmatrix}\mathbf{A}' & \mathbf{D}' & \mathbf{B}' & \mathbf{C}'\end{bmatrix}, \qquad \mathbf{W}' := \begin{bmatrix}\mathbf{A}' & \mathbf{C}' & \mathbf{D}' & \mathbf{B}'\end{bmatrix},$$

where $\mathbf{A}' \in \mathbb{R}^{I^4 \times S'}$ *and* $\mathbf{B}', \mathbf{C}', \mathbf{D}' \in \mathbb{R}^{I^4 \times T'}$, *and, more specifically,*

$$\mathbf{A}'_s := \mathbf{A}_{s_1} \otimes \mathbf{A}_{s_2}, \qquad s := s_2 + (s_1 - 1)S, \qquad s_1, s_2 = 1, \ldots, S, \tag{14}$$

and

$$\mathbf{B}' := \begin{bmatrix}\mathbf{B}'_1 & \mathbf{B}'_2\end{bmatrix}, \qquad \mathbf{C}' := \begin{bmatrix}\mathbf{C}'_1 & \mathbf{C}'_2\end{bmatrix}, \qquad \mathbf{D}' := \begin{bmatrix}\mathbf{D}'_1 & \mathbf{D}'_2\end{bmatrix}, \tag{15}$$

where

$$\begin{aligned}
\mathbf{B}'_{1,t_1} &:= \mathbf{A}_{s_1} \otimes \mathbf{B}_t, & \mathbf{C}'_{1,t_1} &:= \mathbf{A}_{s_1} \otimes \mathbf{C}_t, & \mathbf{D}'_{1,t_1} &:= \mathbf{A}_{s_1} \otimes \mathbf{D}_t, \\
\mathbf{B}'_{2,t_2} &:= \mathbf{B}_t \otimes \mathbf{U}_r, & \mathbf{C}'_{2,t_2} &:= \mathbf{C}_t \otimes \mathbf{W}_r, & \mathbf{D}'_{2,t_2} &:= \mathbf{D}_t \otimes \mathbf{V}_r,
\end{aligned} \tag{16}$$

where $t_1 := t + (s_1 - 1)T$ *and* $t_2 := r + (t - 1)R$, *for* $t = 1, \ldots, T$ *and* $r = 1, \ldots, R$.

Proof. From Proposition 2 follows that $\mathbf{U}'_{i'} = \mathbf{U}_{i_1} \otimes \mathbf{U}_{i_2}$, $\mathbf{V}'_{i'} = \mathbf{V}_{i_1} \otimes \mathbf{V}_{i_2}$, and $\mathbf{W}'_{i'} = \mathbf{W}_{i_1} \otimes \mathbf{W}_{i_2}$, where $i' := i_2 + (i_1 - 1)R$. Thus, the symmetric part of the recursive PD must

satisfy: $\mathbf{U}_{i_1} = \mathbf{V}_{i_1} = \mathbf{W}_{i_1}$ and $\mathbf{U}_{i_2} = \mathbf{V}_{i_2} = \mathbf{W}_{i_2}$. Consequently, i_1 and i_2 must be smaller than or equal to S, such that $\mathbf{U}_{i_1} = \mathbf{V}_{i_1} = \mathbf{W}_{i_1} = \mathbf{A}_{i_1}$ and $\mathbf{U}_{i_2} = \mathbf{V}_{i_2} = \mathbf{W}_{i_2} = \mathbf{A}_{i_2}$. For the other values of i_1 and i_2, the columns still appear in CS pairs and they can be rearranged as in (16) to satisfy the CS structure. □

Note that, since $\text{PD}_{\text{Strassen}}$ satisfies $S = 1$ and $T = 2$, the parameters of $\text{PD}^{\text{rec}}_{\text{Strassen}}$ are $S = 1$ and $T = 16$. Another CS $\text{PD}_{49}(\mathcal{T}_{444})$ is obtained with $(S, T) = (16, 11)$ by using a $\text{PD}_7(\mathcal{T}_{222})$ with $(S, T) = (4, 1)$ recursively.

3. Generalization cyclic symmetry

In this section, a generalization of the CS structure discussed in Section 2.3.2 to the case when I, J, and K are not equal, is proposed. We do this to reduce the number of variables and reduce the search space to speed up the convergence. The basis of the idea is the observation that a MMT for non-square matrix multiplication is still CS as long as the indices in (4) are smaller than the smallest dimension. However, because of the interaction of these CS indices with the non-symmetric indices in a CPD of the tensor, imposing this structure on all rank-1 tensors would be too restrictive. Hence, some unstructured terms are added as well. In Section 4, we give experimental results for the maximum number of structured rank-1 terms for different problem sizes. Because, e.g., a PD of $\mathcal{T}_{KJI}$ can be obtained from a PD of $\mathcal{T}_{IKJ}$, we assume in this chapter that $I \leq K \leq J$. It is known from (4) that:

$$\begin{aligned}&\mathcal{T}_{IKJ}(i_1 + (k_2 - 1)I, k_1 + (j_2 - 1)K, j_1 + (i_2 - 1)J) \\ &= \sum_{i=1}^{r} \mathbf{U}_i(i_1, k_2)\mathbf{V}_i(k_1, j_2)\mathbf{W}_i(j_1, i_2) = \begin{cases} 1 & \text{if } i_1 = i_2, j_1 = j_2, k_1 = k_2 \\ 0 & \text{else} \end{cases},\end{aligned}$$

for all $i_1, i_2 = 1, \ldots, I$, $j_1, j_2 = 1, \ldots, J$, and $k_1, k_2 = 1, \ldots, K$. Consequently, as long as $i_1, i_2, j_1, j_2, k_1, k_2 \leq I$, it holds that

$$\begin{aligned}&\mathcal{T}_{IKJ}(i_1 + (k_2 - 1)I, k_1 + (j_2 - 1)K, j_1 + (i_2 - 1)J) \\ &= \mathcal{T}_{IKJ}(k_1 + (j_2 - 1)I, j_1 + (i_2 - 1)K, i_1 + (k_2 - 1)J) \\ &= \mathcal{T}_{IKJ}(j_1 + (i_2 - 1)I, i_1 + (k_2 - 1)K, k_1 + (j_2 - 1)J).\end{aligned}$$

Therefore,

$$\begin{aligned}\sum_{r=1}^{R} \mathbf{U}_r(i_1, k_2)\mathbf{V}_r(k_1, j_2)\mathbf{W}_r(j_1, i_2) &= \sum_{r=1}^{R} \mathbf{U}_r(k_1, j_2)\mathbf{V}_r(j_1, i_2)\mathbf{W}_r(i_1, k_2) \\ &= \sum_{r=1}^{R} \mathbf{U}_r(j_1, i_2)\mathbf{V}_r(i_1, k_2)\mathbf{W}_r(k_1, j_2),\end{aligned}$$

which can be used to include a CS structure in a subpart of the factor matrices. An illustration is shown in Section 3, where the following submatrices are defined as a function of the position in the factor matrices:

$$\mathbf{U}_r(i_1,k_2) =: \begin{cases} \mathbf{A}_r(i_1,k_2) & \text{if } k_2 \le I, r \le S \\ \mathbf{B}_{r-S}(i_1,k_2) & \text{if } k_2 \le I, S < r \le S+T \\ \mathbf{C}_{r-S-T}(i_1,k_2) & \text{if } k_2 \le I, S+T < r \le S+2T \\ \mathbf{D}_{r-S-2T}(i_1,k_2) & \text{if } k_2 \le I, S+2T < r \le R_{\text{CS}} \\ \tilde{\mathbf{U}}_r(i_1,k_2-I) & \text{if } k_2 > I, r \le R_{\text{CS}} \\ \dot{\mathbf{U}}_{r-R_{\text{CS}}}(i_1,k_2) & \text{if } R_{\text{CS}} < r \end{cases},$$

where $R_{\text{CS}} := S+3T$, and for $r = 1,\dots,R$, $i_1 = 1,\dots,I$, and $k_2 = 1,\dots,K$,

$$\mathbf{V}_r(k_1,j_2) =: \begin{cases} \mathbf{A}_r(k_1,j_2) & \text{if } k_1, j_2 \le I, r \le S \\ \mathbf{D}_{r-S}(k_1,j_2) & \text{if } k_1, j_2 \le I, S < r \le S+T \\ \mathbf{B}_{r-S-T}(k_1,j_2) & \text{if } k_1, j_2 \le I, S+T < r \le S+2T \\ \mathbf{C}_{r-S-2T}(k_1,j_2) & \text{if } k_1, j_2 \le I, S+2T < r \le R_{\text{CS}} \\ \hat{\mathbf{V}}_r(k_1-I,j_2) & \text{if } k_1 > I, j_2 \le I, r \le R_{\text{CS}} \\ \tilde{\mathbf{V}}_r(k_1,j_2-I) & \text{if } j_2 > I, r \le R_{\text{CS}} \\ \dot{\mathbf{V}}_{r-R_{\text{CS}}}(k_1,j_2) & \text{if } R_{\text{CS}} < r \end{cases},$$

for $k_1 = 1,\dots,K$, and $j_2 = 1,\dots,J$, and

$$\mathbf{W}_r(j_1,i_2) =: \begin{cases} \mathbf{A}_r(j_1,i_2) & \text{if } j_1 \le I, r \le S \\ \mathbf{C}_{r-S}(j_1,i_2) & \text{if } j_1 \le I, S < r \le S+T \\ \mathbf{D}_{r-S-T}(j_1,i_2) & \text{if } j_1 \le I, S+T < r \le S+2T \\ \mathbf{B}_{r-S-2T}(j_1,i_2) & \text{if } j_1 \le I, S+2T < r \le R_{\text{CS}} \\ \hat{\mathbf{W}}_r(j_1-I,i_2) & \text{if } j_1 > I, r \le R_{\text{CS}} \\ \dot{\mathbf{W}}_{r-R_{\text{CS}}}(j_1,i_2) & \text{if } R_{\text{CS}} < r \end{cases},$$

for $j_1 = 1,\dots,J$, and $i_2 = 1,\dots,I$, where $\mathbf{A}_s := \text{reshape}(\mathbf{a}_s, I \times I)$, for $s = 1,\dots,S$, $\mathbf{B}_t := \text{reshape}(\mathbf{b}_t, I \times I)$, for $t = 1,\dots,T$, and similarly for $\mathbf{C}_t$ and $\mathbf{D}_t$, and

$$\begin{aligned} \tilde{\mathbf{U}}_r &:= \text{reshape}\big(\tilde{\mathbf{U}}(:,r),\ I \times (K-I)\big), \\ \tilde{\mathbf{V}}_r &:= \text{reshape}\big(\tilde{\mathbf{V}}(:,r),\ K \times (J-I)\big), \\ \hat{\mathbf{V}}_r &:= \text{reshape}\big(\hat{\mathbf{V}}(:,r),\ (K-I) \times I\big), \\ \hat{\mathbf{W}}_r &:= \text{reshape}\big(\hat{\mathbf{W}}(:,r),\ (J-I) \times I\big), \end{aligned}$$

for $r = 1,\dots,R_{\text{CS}}$, and

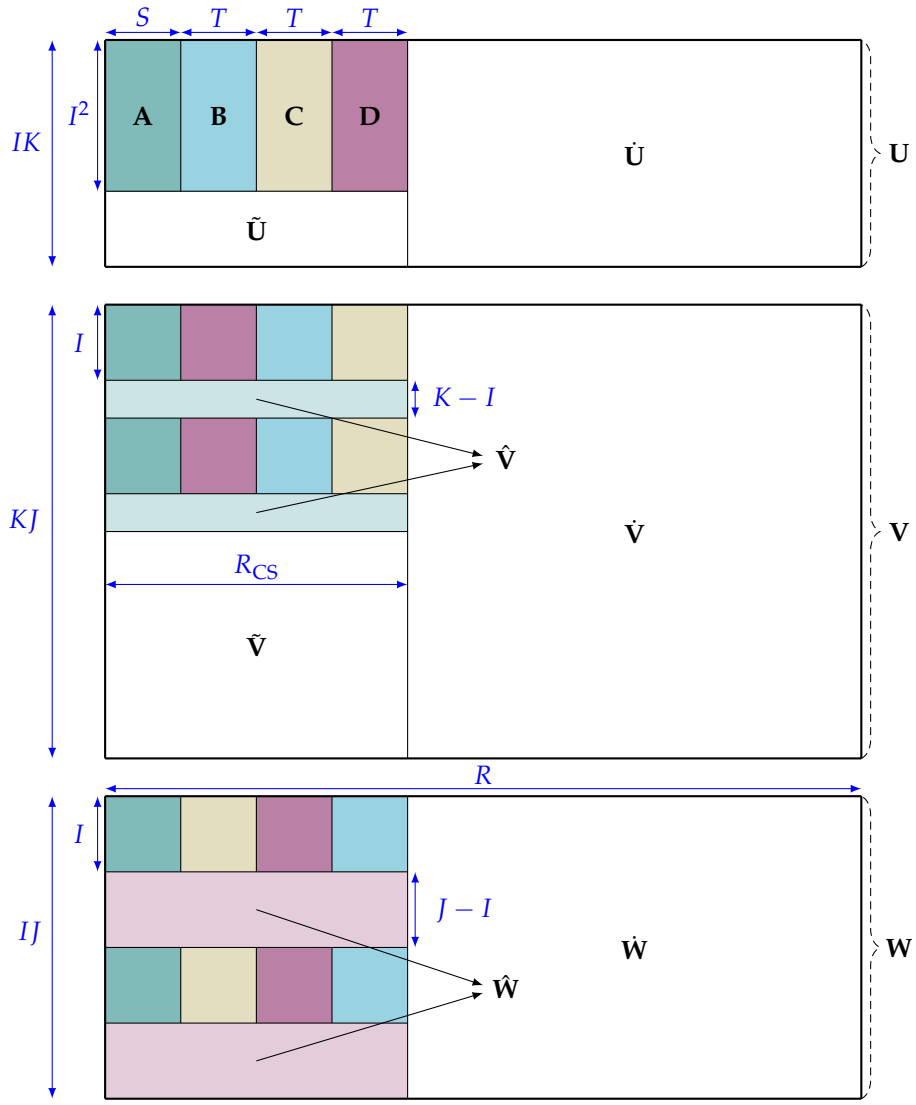

Figure 1. Illustration of the generalized CS factor matrices of a $\mathrm{PD}_{20}(\mathcal{T}_{234})$ with $(S,T) := (2,2)$ and $I = 2$, $K = 3$, and $J = 4$.

$$\begin{aligned}\dot{\mathbf{U}}_r &:= \mathrm{reshape}\big(\dot{\mathbf{U}}(:,r),\ I \times K\big),\\ \dot{\mathbf{V}}_r &:= \mathrm{reshape}\big(\dot{\mathbf{V}}(:,r),\ K \times J\big),\\ \dot{\mathbf{W}}_r &:= \mathrm{reshape}\big(\dot{\mathbf{W}}(:,r),\ J \times I\big),\end{aligned}$$

for $r = 1,\ldots,(R - R_{\mathrm{CS}})$, and thus $\tilde{\mathbf{U}} \in \mathbb{R}^{I(K-I)\times R_{\mathrm{CS}}}$, $\tilde{\mathbf{V}} \in \mathbb{R}^{K(J-I)\times R_{\mathrm{CS}}}$, $\hat{\mathbf{V}} \in \mathbb{R}^{(K-I)I\times R_{\mathrm{CS}}}$, $\hat{\mathbf{W}} \in \mathbb{R}^{(J-I)I\times R_{\mathrm{CS}}}$, $\dot{\mathbf{U}} \in \mathbb{R}^{IK\times(R-R_{\mathrm{CS}})}$, $\dot{\mathbf{V}} \in \mathbb{R}^{KJ\times(R-R_{\mathrm{CS}})}$, and $\dot{\mathbf{W}} \in \mathbb{R}^{IJ\times(R-R_{\mathrm{CS}})}$. The proportions in Section 3 are those of $\mathcal{T}_{234}$, $(S,T) := (2,2)$, and $R := 20$. We show in the experiments that such PDs indeed exist. Note that, contrary to the case when I, J, and K are equal, the CS rank R_{CS} does not equal the rank R, because this would be too restrictive. Also, in the experiments that follow, we did not find PDs for which both are equal. This is likely because of the interaction of the CS part with the parts $\tilde{\mathbf{U}}$, $\tilde{\mathbf{V}}$, $\hat{\mathbf{V}}$, and $\hat{\mathbf{W}}$. However, including this structure still reduces the number of parameters from $(IK + KJ + IJ)R$ to $(IK + KJ + IJ)R - 2I^2R_{\mathrm{CS}}$.

Example 4. *The optimal CPD (in terms of stability and number of nonzeros) for $\mathcal{T}_{223}$ is an extension of Strassen's decomposition [1,39]:*

$$\mathbf{U} = \left[\begin{array}{cccccccccccc} 1 & 1 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \end{array}\right],$$

$$\mathbf{V} = \left[\begin{array}{ccccccccccc} 1 & 0 & -1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{array}\right],$$

$$\mathbf{W} = \left[\begin{array}{ccccccccccc} 1 & 0 & 1 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{array}\right].$$

The CS part can clearly be recognized. Remark that the matrices $\tilde{\mathbf{V}}$ and $\hat{\mathbf{W}}$ are zero in this case. However, this is not always the case, as shown for example in Section 4. Similarly, the optimal $\mathrm{PD}_{14}(\mathcal{T}_{224})$s and $\mathrm{PD}_{18}(\mathcal{T}_{225})$s can also be written with $(S,T) = (1,2)$.

3.1. Computational complexity optimization

The most computationally expensive step in a Gauss–Newton-type method, such as the LM method used in Section 4, is solving the linear system for the step, which requires $\mathcal{O}\left((R(IK+KJ+IJ))^3\right)$ operations [40]. By incorporating the generalized CS structure, the complexity is directly reduced to $\mathcal{O}\left(\left(R(IK+KJ+IJ)-2I^2R_{\mathrm{CS}}\right)^3\right)$ operations. For square matrix multiplication and when $R_{\mathrm{CS}} = R$, the complexity is reduced by a factor of 3^3.

3.2. Complexity of multiplying a matrix with itself

When multiplying a matrix with itself, i.e., $\mathbf{Z} := \mathbf{XX}$, the CS structure can be exploited to save computations.

Proposition 5. *Let $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{W}$ be the factor matrices of a base algorithm for (I,I,I) matrix multiplication. When $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{W}$ are CS with parameters S and T, and $R_{\mathrm{CS}} = R$, and $\mathbf{Z} := \mathbf{XX}$, with $\mathbf{X} \in \mathbb{R}^{I^l \times I^l}$, then the computational cost of applying the base algorithm l times recursively derived in Proposition 1 is reduced to*

$$\mathrm{cost}_{\mathrm{CS}}\left(I^l, I^l, I^l\right) = 2\left(1+\frac{c_{\mathbf{U}}}{\delta}\right)R^l - (2c_{\mathbf{U}}+\delta)\frac{I^{2l}}{\delta} \tag{17}$$

$$= \mathrm{cost}\left(I^l, I^l, I^l\right) - c_{\mathbf{U}}\left(\frac{R^l - I^{2l}}{\delta}\right), \tag{18}$$

where $\delta := R - I^2 > 0$ and $c_{\mathbf{U}}$ is the number of additions and scalar multiplications needed to compute the inner products with $\mathbf{u}_r$ in (6).

Proof. When $\mathbf{Y} = \mathbf{X}$ in base algorithm (6) and when the decomposition is CS with $R_{\text{CS}} = R$, the inner product with $\mathbf{v}_r$ are the same as those with $\mathbf{u}_r$ but in a different order:

$$\begin{aligned}\text{vec}\left(\mathbf{Z}^\top\right) &= \sum_{r=1}^{R} \langle \mathbf{u}_r, \text{vec}(\mathbf{X})\rangle \langle \mathbf{v}_r, \text{vec}(\mathbf{X})\rangle \mathbf{w}_r \\ &= \sum_{s=1}^{S} \langle \mathbf{a}_s, \text{vec}(\mathbf{X})\rangle \langle \mathbf{a}_s, \text{vec}(\mathbf{X})\rangle \mathbf{a}_s + \sum_{t=1}^{T} \Big(\langle \mathbf{b}_t, \text{vec}(\mathbf{X})\rangle \langle \mathbf{d}_t, \text{vec}(\mathbf{X})\rangle \mathbf{c}_t \ldots \\ &\quad \langle \mathbf{c}_t, \text{vec}(\mathbf{X})\rangle \langle \mathbf{b}_t, \text{vec}(\mathbf{X})\rangle \mathbf{d}_t + \langle \mathbf{d}_t, \text{vec}(\mathbf{X})\rangle \langle \mathbf{c}_t, \text{vec}(\mathbf{X})\rangle \mathbf{b}_t \Big).\end{aligned}$$

Consequently, the recursive formula for the cost is

$$\text{cost}\left(I^l, I^l, I^l\right) = R \cdot \text{cost}\left(I^{l-1}, I^{l-1}, I^{l-1}\right) + c_{\mathbf{U}} \cdot \left(I^2\right)^{l-1} + c_{\mathbf{W}} \cdot \left(I^2\right)^{l-1}.$$

Because of the CS, the elements in $\mathbf{W}$ are the same as those of $\mathbf{U}$ in a different order. Consequently, $c_{\mathbf{W}} = c_{\mathbf{U}} + R - I^2$. The recursive formula can be expanded in the same way as in the proof of Proposition 1 to obtain:

$$\begin{aligned}\text{cost}_{\text{CS}}\left(I^l, I^l, I^l\right) &= \left(1 + c_{\mathbf{U}} \frac{1}{R - I^2} + c_{\mathbf{W}} \frac{1}{R - I^2}\right) R^l - c_{\mathbf{U}} \frac{I^{2l}}{R - I^2} - c_{\mathbf{W}} \frac{I^{2l}}{R - I^2} \\ &= \text{cost}\left(I^l, I^l, I^l\right) - c_{\mathbf{V}} \frac{R^l}{R - I^2} + c_{\mathbf{V}} \frac{I^{2l}}{R - I^2} \\ &= \text{cost}\left(I^l, I^l, I^l\right) - c_{\mathbf{U}} \left(\frac{R^l - I^{2l}}{R - I^2}\right) \\ &= \left(1 + \left(2c_{\mathbf{U}} + R - I^2\right) \frac{1}{R - I^2}\right) R^l - \left(2c_{\mathbf{U}} + R - I^2\right) \frac{I^{2l}}{R - I^2} \\ &= 2\left(1 + c_{\mathbf{U}} \frac{1}{R - I^2}\right) R^l - \left(2c_{\mathbf{U}} + R - I^2\right) \frac{I^{2l}}{R - I^2}.\end{aligned}$$

□

Example 6 (Complexity Strassen with and without CS). *Strassen's decomposition for $I = 2$ has rank 7 and 12 nonzeros in each factor matrix. Thus $c_{\mathbf{U}} = c_{\mathbf{V}} = 12 - R = 5$, and $c_{\mathbf{W}} = 12 - I^2 = 8$. The number of operations required as a function of the number of recursive levels l is shown in Figure 2. When exploiting the CS in Strassen's decomposition, the algorithm improves standard matrix multiplication for the multiplication of a matrix with itself already after 8 recursive levels, thus for matrices of size $2^8 = 256$ or higher. Without CS, there is an improvement over standard matrix multiplication only from size $2^{10} = 1024$, and the improvement is less significant.*

4. Numerical experiments

This section presents the results obtained using the AL method [3] to find PDs of MMTs with the generalized CS structure, for various matrix sizes (I, K, J) and different CS parameters S and T. The rank is always chosen as the lowest rank $\tilde{R}$ for which a solution is known to exist in the literature. For the first two experiments, this is also known to be the optimal one [28,41].

The AL method requires upper and lower bounds on the elements in the decomposition: $l \leq \mathbf{x} \leq u$. We set $u := -l := 1$ and generate 50 random starting points of magnitude 10^{-2} using the built-in `randn` function in MATLAB, for each set of problem parameters:

$$\mathbf{x}_0 := 10^{-2} \cdot \texttt{randn}\Big((IK + KJ + IJ)R - 2I^2 R_{\text{CS}} \times 1\Big). \tag{19}$$

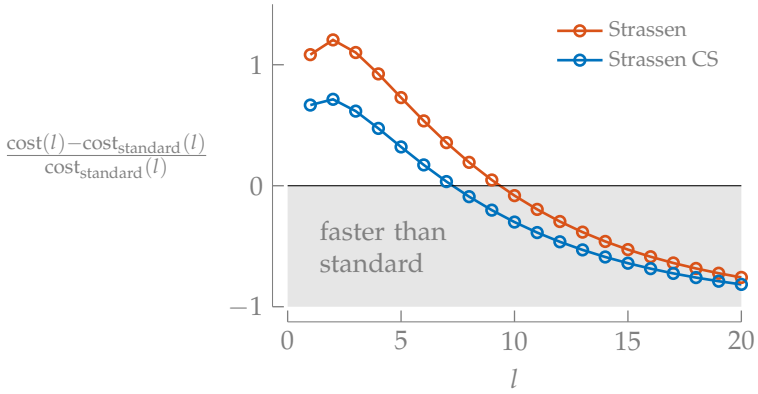

Figure 2. Relative improvement compared to standard matrix multiplication of the operations required to multiply a matrix with itself with (blue) and without (red) CS in $\text{PD}_{\text{Strassen}}$.

The number of inner iterations of the LM method was set to 50, and the number of outer iterations to 15. The tolerances on the gradient and the constraint were set to 10^{-13}.

When a numerical solution with a cost function value smaller than 10^{-12} is found, the constraint h_{discr} from (10), scaled by a factor of 0.1, is added to the optimization problem to convert the numerical solution into a practical one. It is known that not all numerical solutions can be converted into practical ones using inv-transformations [36]. Although we do not use inv-transformations but rather numerical optimization, we also observed that many numerical solutions do not converge to a discrete solution under this constraint, while others converge very quickly. We verified that many of the non-converging numerical solutions also fail to satisfy the necessary conditions for discretizability from [36].

The rank of the Jacobian matrix $\mathbf{J}_{\text{CS,gen}}$ of the generalized structure at the discrete solutions is computed. The rank deficiency reflects the dimension of the invariance transformation applicable to the decomposition. Two decompositions with different ranks cannot be transformed into one another using inv-transformations [3]. Thus, the more distinct ranks that are found, the more 'unique' solutions are discovered.

The MMTs are ordered according to their best-known rank in Table 1. The following paragraphs present the results for sizes between $(2,2,2)$ and $(3,3,3)$, both corresponding to square matrix multiplication tensors. Larger sizes are possible, but at a certain point, the computation time per experiment becomes too large to perform all 50 experiments per parameter set in a reasonable amount of time on a standard computer.

Table 1. The best known ranks for different MMT sizes. The yellow colored problems are tested.

(I,K,J)	$\tilde{R}$
$(2,2,2)$	7
$(2,2,3)$	11
$(2,2,4)$	14
$(2,3,3)$	15
$(2,2,5)$	18
$(2,3,4)$	20
$(3,3,3)$	23
⋮	⋮

All experiments were performed on a laptop with an AMD Ryzen 7 PRO 7840U processor and 64 GB of RAM, using Tensorlab 4 beta [42] and Matlab R2023b. The results

of the experiments, including the new decompositions, as well as the code, are publicly available[1].

Problem 1: $(I, K, J) = (2, 2, 3), R := R^* = 11$

Table 2 presents the results for $\mathrm{PD}_{11}(\mathcal{T}_{223})$s for different combinations of S and T. The rightmost column indicates the number of practical solutions found out of 50 random starting points. The bottom row shows the results without imposing any structure.

The size, minimal and maximal rank, and the number of distinct ranks of $\mathbf{J}_{\mathrm{CS,gen}}$ are provided in the three middle columns. For a complete list of all distinct ranks, we refer to the detailed results available online. As expected, the second dimension of the Jacobian matrix and its rank decrease as R_{CS} increases, due to the reduced parameter space. Also the null-space dimension of the Gauss–Newton Gramian $\mathbf{J}_{\mathrm{CS,gen}}^{\top}\mathbf{J}_{\mathrm{CS,gen}}$ reduces with R_{CS}.

From Table 2, it is evident that significantly more practical solutions are found for moderate values of S and T compared to the unstructured case (bottom row). Specifically, for $(S, T) = (3, 1)$ and $(S, T) = (1, 1)$, the number of practical solutions more than doubles. Additionally, for $(S, T) = (1, 1)$, the number of unique solutions also doubles. All improvements are highlighted in bold.

Table 2. The number of (unique) practical PDs of rank 11 for $\mathcal{T}_{223}$ increases significantly for moderate values of R_{CS}. Improvements relative to the unstructured case (bottom row) are highlighted in bold.

R_{CS}	S	T	size($\mathbf{J}_{\mathrm{CS,gen}}$)	rank($\mathbf{J}_{\mathrm{CS,gen}}(\mathbf{x}^*)$) min	max	#	# $\mathbf{x}^*$ pract.
8	2	2	144×112	92	97	1	1
7	1	2	144×120	85	90	4	10
	4	1		87	91	4	**18**
6	0	2	144×128	91	105	**7**	13
	3	1		93	100	**8**	**36**
5	2	1	144×136	99	117	**8**	**25**
4	1	1	144×144	105	123	**13**	**30**
	4	0		107	118	**10**	**18**
3	0	1	144×152	111	128	**11**	**18**
	3	0		113	121	**8**	**29**
2	2	0	144×160	119	127	**8**	**21**
1	1	0	144×168	123	131	**7**	14
0	0	0	144×176	125	132	6	15

Problem 2: $(I, K, J) = (2, 2, 4), R := R^* = 14$

The results for $\mathrm{PD}_{14}(\mathcal{T}_{224})$ are shown in Table 3, again for different values of R_{CS}, S, and T. The largest CS rank obtained is $R_{\mathrm{CS}} = 7$. The largest number of practical solutions is found for $(S, T) = (3, 1)$, which is four times higher than the number without structure ($(S, T) = (0, 0)$). Also here, it is evident that including the generalized CS structure leads to more practical PDs and also yields a greater variety of Jacobian ranks, indicating more non-inv-equivalent PDs.

[1] https://github.com/CharlotteVermeylen/ALM_FMM_gen_CS

Table 3. The number of (unique) practical PDs of rank 14 for $\mathcal{T}_{224}$ increases with R_{CS} for almost all combinations of S and T. In particular, for $(S,T)=(3,1)$, the number of practical decompositions is four times higher. Improvements relative to the unstructured case are indicated in bold.

R_{CS}	S	T	size$(\mathbf{J}_{\text{CS,gen}})$	rank$\big(\mathbf{J}_{\text{CS,gen}}(\mathbf{x}^*)\big)$ min	max	#	# $\mathbf{x}^*$ pract.
7	1	2	256×224	178	182	3	**9**
	4	1		180	186	**4**	**14**
6	0	2	256×232	184	191	**5**	**9**
	3	1		184	190	**4**	**21**
5	2	1	256×240	188	201	**7**	**16**
4	1	1	256×248	196	204	**7**	**12**
	4	0		196	210	**9**	**12**
3	0	1	256×256	202	209	**4**	4
	3	0		202	212	**6**	**11**
2	2	0	256×264	214	220	2	4
1	1	0	256×272	217	218	2	4
0	0	0	256×280	217	220	3	5

Problem 3: $(I,K,J)=(2,3,3), R:=\tilde{R}=15$

In Table 4, the results for $\mathcal{T}_{233}$ at rank 15 are presented. The maximum value of R_{CS} obtained is 9. The same conclusions as in the previous sections apply; however, generally fewer practical solutions are found because the problem becomes increasingly difficult as I, K, and J grow.

Table 4. The number of (unique) practical PDs of rank 15 of $\mathcal{T}_{233}$ increases with R_{CS} for all values of S and T. Without CS structure, no practical decompositions are found. Improvements w.r.t. the unstructured case (bottom row) are indicated in bold.

R_{CS}	S	T	size$(\mathbf{J}_{\text{CS,gen}})$	rank$\big(\mathbf{J}_{\text{CS,gen}}(\mathbf{x}^*)\big)$ min	max	#	# $\mathbf{x}^*$ pract.
9	6	1	324×243	218	218	**1**	**1**
8	5	1	324×251	226	226	**1**	**2**
7	1	2	324×259	238	228	**1**	**2**
	4	1		230	234	**3**	**5**
6	0	2	324×267	324	234	**1**	**1**
	3	1		236	238	**2**	**10**
	6	0		238	238	**1**	**1**
5	2	1	324×275	242	242	**1**	**7**
4	1	1	324×283	248	248	**1**	**2**
	4	0		250	254	**4**	**6**
3	0	1	324×291	254	254	**1**	**2**
	3	0		254	259	**4**	**6**
2	2	0	324×299	258	261	**3**	**4**
0	0	0	324×315	–	–	0	0

The following decomposition has the same number of nonzeros and stability parameters as the algorithm proposed in [35] but with CS parameters $(S, T) = (1, 2)$:

$$\mathbf{U} = \left[\begin{array}{cccccccc|cccccccc} 1 & 0 & 0 & -1 & -1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 1 \\ 1 & -1 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 1 & 0 & -1 & 1 & 0 \end{array}\right],$$

$$\mathbf{V} = \left[\begin{array}{cccccccc|cccccccc} 1 & 0 & 1 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \end{array}\right],$$

$$\mathbf{W} = \left[\begin{array}{cccccccc|cccccccc} 1 & -1 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & -1 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & -1 & -1 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & -1 & 0 & -1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 1 & 0 & 1 & 1 \end{array}\right].$$

In the CS part, Strassen's decomposition, which also satisfies $(S, T) = (1, 2)$, can be recognized. To our knowledge, this is one of the optimal known algorithms for $(I, J, K) = (2, 3, 3)$, or any permutation of the dimensions.

Problem 4: $(I, K, J) = (2, 2, 5), R := \tilde{R} = 18$

Table 5 presents the results for $\mathcal{T}_{225}$ at rank 18. The maximal CS rank is 9, as was the case in Table 4. Notably, for $(S, T) = (3, 1)$, ten times more practical solutions are found compared to the unstructured case.

Table 5. The number of (unique) practical $\mathrm{PD}_{18}(\mathcal{T}_{225})$s increases with R_{CS} for almost all combinations of S and T. For $(S,T)=(3,1)$, ten times more practical solutions are found compared to $(S,T)=(0,0)$. The improvements are indicated in bold.

R_{CS}	S	T	size$(\mathbf{J}_{\mathrm{CS,gen}})$	rank$\big(\mathbf{J}_{\mathrm{CS,gen}}(\mathbf{x}^*)\big)$ min	max	#	# $\mathbf{x}^*$ pract.
9	6	1	400×360	288	288	1	1
7	1	2	400×376	292	299	**4**	**4**
	4	1		293	301	**3**	**3**
6	0	2	400×384	297	306	**3**	**3**
	3	1		299	310	**8**	**10**
5	2	1	400×392	305	312	**8**	**8**
4	1	1	400×400	313	325	**4**	**4**
	4	0		312	323	**5**	**5**
3	0	1	400×408	327	327	1	1
	3	0		322	327	**4**	**5**
2	2	0	400×416	330	338	**2**	**2**
1	1	0	400×424	332	332	1	**2**
0	0	0	400×432	336	336	1	1

Problem 5: $(I,K,J)=(2,3,4), R:=\tilde{R}=20$

Lastly, Table 6 presents the results for $\mathcal{T}_{234}$ at rank 20. Very few practical solutions are found in this case. Electronic versions of the decompositions are available. When using 1000 starting points for $S=4$ and $T=1$, the most sparse practical PD that we found has 144 nonzeros and stability factors $Q=16$ and $E=57$, which is slightly higher than the decomposition proposed in [8], which has 130 nonzeros and $Q=14$ and $E=35$.

Table 6. Practical $\mathrm{PD}_{20}(\mathcal{T}_{234})$s are found for four combinations of S and T. Without CS structure ($(S,T)=(0,0)$), no practical decompositions are found.

R_{CS}	S	T	size$(\mathbf{J}_{\mathrm{CS,gen}})$	rank$\big(\mathbf{J}_{\mathrm{CS,gen}}(\mathbf{x}^*)\big)$	$\mathbf{x}^*$ pract.
8	5	1	576×456	408	**1**
7	4	1	576×464	410	**1**
5	2	1	576×480	425	**1**
4	4	0	576×488	427	**1**
0	0	0	576×520	-	0

Remark: square matrix multiplication

Note that the CS rank can also be chosen smaller than R when $I=J=K$. For example, 11 symmetric rank-1 tensors can be enforced within a $\mathrm{PD}_{23}(\mathcal{T}_{333})$. The remaining 12 rank-1 tensors do not need to follow any specific structure in this case. To our knowledge, no decomposition with this many symmetric terms has been reported in the literature for $\mathcal{T}_{333}$. An example decomposition with this structure is available in electronic version.

5. Conclusion

The proposed generalized cyclic symmetric structure can be used in polyadic decompositions of the fast matrix multiplication (FMM) tensor to reduce the number of parameters. This reduction decreases the size of the search space and shortens the computation time per iteration of an optimization algorithm to find new decompositions. We have shown that the CS structure can be exploited to reduce the computational cost of multiplying a matrix with itself. For several problem parameters—i.e., different matrix multiplication sizes—numerical experiments using a state-of-the-art optimization algorithm demonstrate that incorporating this structure enables significantly more practical solutions to be found

for a fixed number of starting points. Although we did not directly improve upon the existing FMM algorithms, the ability to find new decompositions more easily can support further research toward faster FMM algorithms, particularly when combined with sufficient computational resources.

Author Contributions: Conceptualization, C.V. and M.V.B.; methodology, C.V.; software, C.V. and N.V.; validation, C.V.; formal analysis, C.V.; investigation, C.V.; resources, M.V.B.; data curation, C.V.; writing—original draft preparation, C.V.; writing—review and editing, C.V., N.V., and M.V.B.; visualization, C.V.; supervision, M.V.B.; project administration, L.D.L. and M.V.B.; funding acquisition, L.D.L and M.V.B. All authors have read and agreed to the published version of the manuscript.

Funding: The following funders supported this work: (1) the Flemish Government, under the AI Research Program. Charlotte Vermeylen and Lieven De Lathauwer are affiliated to Leuven.AI - KU Leuven institute for AI, B-3000, Leuven, Belgium. (2) KU Leuven Internal Funds: iBOF/23/064, and C14/22/096. Nico Vervliet holds a research grant from the Research Foundation Flanders (FWO) (12ZM223N). Marc Van Barel was supported by the Research Council KU Leuven, C1-project C14/17/073 and by the Fund for Scientific Research–Flanders (Belgium), EOS Project no 30468160 and project G0B0123N.

Data Availability Statement: The results of the experiments, including the new decompositions, as well as the code of the optimization algorithm are publicly available at https://github.com/CharlotteVermeylen/ALM_FMM_gen_CS.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

AL	Augmented Lagrangian
BRPD	Border Rank Polyadic Decomposition
(C)PD	(Canonical) Polyadic Decomposition
CS	Cyclic Symmetry
FMM	Fast Matrix Multiplication
LM	Levenberg–Marquardt
MMT	Matrix Multiplication Tensor
NLS	Nonlinear least squares

References

1. Strassen, V. Gaussian elimination is not optimal. *Numerische Mathematik* **1969**, *13*, 354–356.
2. Fawzi, A.; Balog, M.; Huang, A.; Hubert, T.; Romera-Paredes, B.; Barekatain, M.; Novikov, A.; R. Ruiz, F.J.; Schrittwieser, J.; Swirszcz, G.; et al. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature* **2022**, *610*, 47–53.
3. Vermeylen, C.; Van Barel, M. Stability improvements for fast matrix multiplication. *Numerical Algorithms* **2024**, *CAM23*.
4. Smirnov, A.V. The bilinear complexity and practical algorithms for matrix multiplication. *Computational Mathematics and Mathematical Physics* **2013**, *53*, 1781–1795.
5. Tichavský, P.; Phan, A.H.; Cichocki, A. Numerical CP decomposition of some difficult tensors. *Journal of Computational and Applied Mathematics* **2017**, *317*, 362–370.
6. Heule, M.J.; Kauers, M.; Seidl, M. New ways to multiply 3×3-matrices. *Journal of Symbolic Computation* **2021**, *104*, 899–916.
7. Ballard, G.; Ikenmeyer, C.; Landsberg, J.M.; Ryder, N. The geometry of rank decompositions of matrix multiplication II: 3×3 matrices. *Journal of Pure and Applied Algebra* **2019**, *223*, 3205–3224.
8. Benson, A.R.; Ballard, G. A framework for practical parallel fast matrix multiplication. In Proceedings of the 20th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, New York, NY, USA, 2015; PPoPP 2015, p. 42–53.
9. Bürgisser, P.; Clausen, M.; Shokrollahi, M.A. *Algebraic complexity theory*; Vol. 315, Springer Science & Business Media, 2013.
10. Landsberg, J.M. *Tensors: Geometry and Applications*; Vol. 128, American Mathematical Soc., 2011.
11. Brent, R.P. Algorithms for matrix multiplication. Technical report, Stanford, CA, USA, 1970.

12. Landsberg, J.M. *Geometry and Complexity Theory*; Cambridge Studies in Advanced Mathematics, Cambridge University Press: Cambrige, 2017.
13. de Groote, H.F. On varieties of optimal algorithms for the computation of bilinear mappings II. Optimal algorithms for 2×2-matrix multiplication. *Theoretical Computer Science* **1978**, *7*, 127–148.
14. Bläser, M. On the complexity of the multiplication of matrices of small formats. *Journal of Complexity* **2003**, *19*, 43–60.
15. Laderman, J.D. A non commutative algorithm for multiplying 3×3 matrices using 23 multiplications. *Bulletin of the American Mathematical Society* **1976**, *82*, 126–128.
16. Novikov, A.; Vũ, N.; Eisenberger, M.; Dupont, E.; Huang, P.S.; Wagner, A.Z.; Shirobokov, S.; Kozlovskii, B.; Ruiz, F.J.R.; Mehrabian, A.; et al. AlphaEvolve: A coding agent for scientific and algorithmic discovery, 2025, [arXiv:cs.AI/2506.13131].
17. Dumas, J.G.; Pernet, C.; Sedoglavic, A. A non-commutative algorithm for multiplying 4x4 matrices using 48 non-complex multiplications, 2025, [arXiv:cs.SC/2506.13242].
18. Kaporin, I. Finding complex-valued solutions of Brent equations using nonlinear least squares. *Computational Mathematics and Mathematical Physics* **2024**, *64*, 1881–1891.
19. Schwartz, O.; Zwecher, E. Towards faster feasible matrix multiplication by trilinear aggregation, 2025, [arXiv:cs.DS/2508.01748].
20. Chiantini, L.; Ikenmeyer, C.; Landsberg, J.; Ottaviani, G. The geometry of rank decompositions of matrix multiplication I: 2×2 matrices. *Experimental Mathematics* **2019**, *28*, 322–327.
21. de Groote, H.F. On varieties of optimal algorithms for the computation of bilinear mappings I. The isotropy group of a bilinear mapping. *Theoretical Computer Science* **1978**, *7*, 1–24.
22. Krijnen, W.P.; Dijkstra, T.K.; Stegeman, A. On the non-existence of optimal solutions and the occurrence of "degeneracy" in the CANDECOMP/PARAFAC model. *Psychometrika* **2008**, *73*, 431–439.
23. Paatero, P. Construction and analysis of degenerate PARAFAC models. *Journal of Chemometrics* **2000**, *14*, 285–299.
24. Bini, D.; Capovani, M.; Romani, F.; Lotti, G. $\mathcal{O}(n^{2.7799})$ complexity for n × n approximate matrix multiplication. *Information Processing Letters* **1979**, *8*.
25. Landsberg, J.M.; Ottaviani, G. New lower bounds for the border rank of matrix multiplication. *Theory of Computing* **2015**, *11*, 285–298.
26. Landsberg, J.; Michatek, M. On the geometry of border rank algorithms for matrix multiplication and other tensors with symmetry. *Siam Applied Algebraic Geometry* **2017**, *1*, 2–19.
27. Schönhage, A. Partial and total matrix multiplication. *SIAM Journal on Computing* **1981**, *10*, 434–455.
28. Alekseev, V.B.; Smirnov, A.V. On the exact and approximate bilinear complexities of multiplication of 4×2 and 2×2 matrices. *Proceedings of the Steklov Institute of Mathematics* **2013**, *282*, 123–139.
29. Gong, X.; Mohlenkamp, M.J.; Young, T.R. The optimization landscape for fitting a rank-2 tensor with a rank-1 tensor. *SIAM Journal on Applied Dynamical Systems* **2018**, *17*, 1432–1477.
30. Mohlenkamp, M.J. The Dynamics of Swamps in the Canonical Tensor Approximation Problem. *SIAM Journal on Applied Dynamical Systems* **2019**, *18*, 1293–1333.
31. Vermeylen, C.; Vervliet, N.; De Lathauwer, L. Reducing swamp behavior for the canonical polyadic decomposition problem by rank-1 freezing. *Numerical Algorithms* **2025-06-05**.
32. Johnson, R.W.; McLoughlin, A.M. Noncommutative bilinear algorithms for 3×3 matrix multiplication. *SIAM Journal on Computing* **1986**, *15*, 595–603.
33. Uschmajew, A. Local convergence of the alternating least squares algorithm for canonical tensor approximation. *SIAM Journal on Matrix Analysis and Applications* **2012**, *33*, 639–652.
34. Nocedal, J.; Wright, S.J. *Numerical Optimization*; Springer: New York, 2006.
35. Ballard, G. Discovering fast matrix multiplication algorithms via tensor decomposition. In Proceedings of the SIAM Conference on Computational Science and Engineering, 2017, Vol. 30.
36. Berger, G.O.; Absil, P.A.; De Lathauwer, L.; Jungers, R.M.; Van Barel, M. Equivalent polyadic decompositions of matrix multiplication tensors. *Journal of Computational and Applied Mathematics* **2022**, *406*, 113941.
37. Huang, J.; Rice, L.; Matthews, D.A.; van de Geijn, R.A. Generating families of practical fast matrix multiplication algorithms. In Proceedings of the 2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2017, pp. 656–667.
38. Vermeylen, C. Tensor Methods: Fast Matrix Multiplication and Rank Adaptation. PhD thesis, 2024-03-29.
39. Ballard, G.; Benson, A.R.; Druinsky, A.; Lipshitz, B.; Schwartz, O. Improving the numerical stability of fast matrix multiplication. *SIAM Journal on Matrix Analysis and Applications* **2016**, *37*, 1382–1418.
40. Sorber, L.; Van Barel, M.; De Lathauwer, L. Optimization-based algorithms for tensor decompositions: Canonical polyadic decomposition, decomposition in rank-$(L_r, L_r, 1)$ terms, and a new generalization. *SIAM Journal on Optimization* **2013**, *23*, 695–720.
41. Alekseyev, V.B. On the complexity of some algorithms of matrix multiplication. *Journal of Algorithms* **1985**, *6*, 71–85.
42. Vervliet, N.; Debals, O.; Sorber, L.; Van Barel, M.; De Lathauwer, L. Tensorlab 3.0, 2016. Available online at https://www.tensorlab.net.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.